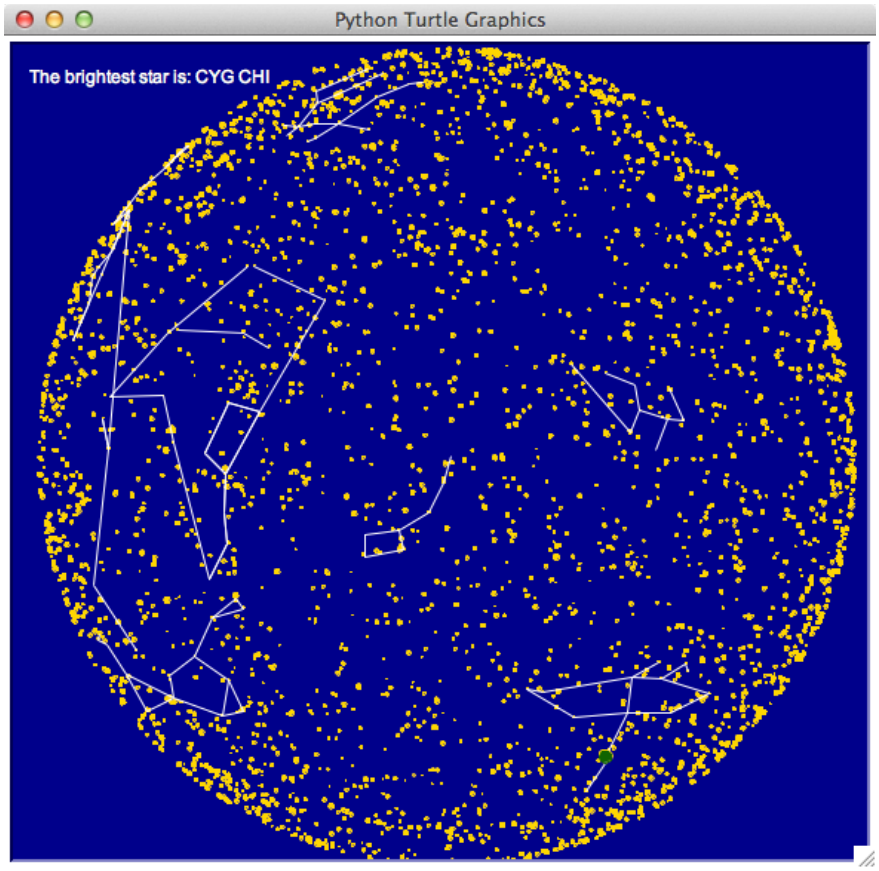
CS110 Project 3: Stars and Constellations.

Prof. Karpenko

Due: Sunday Nov 16th, 2014, 11:59pm

For project 3 you will write a program that displays real astronomical data in 2D: a large set of stars and several constellations. You will add all stars that have a name to a "dictionary" where each key is the name of the star, and the value is a tuple (x, y), the coordinates of the star. Your program will also display a text in the top left corner saying which star is the brightest and highlight it. In the image below, stars are shown as yellow circles, where the radius of each star is proportional to its brightness. Stars that are connected by white lines belong to the same constellation. You will be using files, dictionaries, and strings for this assignment. [1]



[1] The assignment is modified from the following assignment (http://nifty.stanford.edu/2009/reid-starmap/). The data files are courtesy of Karen Reid.

Data Files

The star data

The star data was provided by Professor Karen Reid, who extracted it from the VizieR Catalog (http://vizier.cfa.harvard.edu/). One of advantages of this assignment is the opportunity to work with real scientific data. Here is one of the lines of the stars.txt file, let's look at it closely:
0.103579 0.994400 -0.020976 37128 1.69 1903 ALNILAM; ORI EPSILON

The first three values are the **x , y and z coordinates** of the star:
0.103579 0.994400 -0.020976 37128 1.69 1903 ALNILAM; ORI EPSILON
We will ignore the z coordinate for the purpose of this assignment, and use only the x and y coordinates. Each axis in the coordinate system goes from -1 to +1, and the center point is 0, 0.

The fourth field is the Henry Draper number, which is a unique identifier for the star; we will not be using it for this project. The fifth field is the brightness of the star. Your program will find the star with the maximum brightness, display its name and highlight it in green. The sixth field is the Harvard Revised number, another identifier; you will not be using it for the project.

The seventh field exists only for a small number of stars and is a semicolon-separated list of names for a star. A star may have several names. In this example, the star has two names: ALNILAM and ORI EPSILON.

0.103579 0.994400 -0.020976 37128 1.69 1903 ALNILAM; ORI EPSILON

The constellation data[2]

We provided you with 8 files that contain constellation data. Each line has the names of two stars that should be connected by a line. For example, here is a line from Gemini_lines.txt:
GEM UPSILON,GEM KAPPA
This line in the file means that GEM UPSILON star should be connected by a line to a GEM KAPPA star because they belong to the same constellation.

Star catalog coordinate system

x and y coordinates of stars in the star catalog coordinate system are in the range from -1 to +1. When you parse each line and extract x, y coordinates, you need to convert them to the screen coordinates that vary from –WIDTH to +WIDTH, where

[2] Constellation data is a courtesy of Prof. Karen Reid who collected it from different sources.

WIDTH is half of the width of the screen (assume that the screen is a square, so height=width). Write a helper function **convertXY(origX, origY, width)** that converts the original x, y coordinates in the file to the screen coordinates of the turtle. It should return a tuple (xScreen, yScreen) .

Drawing a star

Write a helper function **drawStar(x, y, t, brightness, col)** that uses the turtle t to display a star at position (x, y) as a circle of color col with the radius proportional to the brightness of the star. This function should take the original x and y coordinates in the star coordinate system and first call convertXY(origX, origY, WIDTH) to get xScreen, yScreen coordinates of the star.

Reading a star file

Write a function called **loadStars(filename, t)** that takes a name of the file that contains star data, and the turtle t and does the following:

- Opens the file
- Reads the lines of the file
- For each line
    - splits it into separate values (using the split function) and extracts x and y coordinates, the brightness of the star and, if the star has name(s), the names of the star.
    - calls drawStar(x, y, t, brightness, "yellow")
    - For any star that has at least one name:
        - For each name of the star, adds a (key, value) pair to the dictionary, where the key is the name of the star, and the value is the (x, y) coordinates of the star.
    - Computes the star with the maximum brightness and displays its name on the turtle screen (using t.write(text) command)
    - Returns this dictionary

Displaying constellations

Write a function **drawConstellations(dictStars, filename, t)** that takes three parameters: the dictionary that maps star names to the (x, y) coordinates ; the name of the file that contains constellation data, and the turtle t. This function should open the file with constellation data, and draw lines between stars that belong to the same constellation. This function should use a helper function drawLine that draws a line between two points (x1, y1) and (x2, y2). The helper function should also take the turtle t as a parameter, and the color of the turtle.

drawLine(x1, y1, x2, y2, t, col)

drawConstellations function should use a dictionary to look up (x,y) coordinates of a star given its name. This is the dictionary that is computed and returned by the loadStars function.

main function

Your program should have a main function that will create the screen and the turtle, and then call loadStars and loadConstellations. Since you have multiple constellation files, you can iterate over all of them using the following syntax:

```
import glob
# The line below returns the list of all the filenames that end with "_lines.txt" in the
# current directory.
listFiles = glob.glob("*_lines.txt")
for filename in listFiles:
        drawConstellations(yourDictionary, filename, yourTurtle)
```

Grading:

This project is worth 10% of your total grade. **You are not allowed to use any code from the web or collaborate on the project with anybody.** You may receive help only from the instructor, the TAs or the CS tutors. I will randomly select several people from each section of cs110, and ask them to come for an interactive code-walkthrough. If you submit the code that you cannot explain to me during the code review, you will get a 0 for the project.

Submission: Save your file in **project3.py** . Thoroughly test your code before submitting it and **add comments to your code**. Upload all your files to Canvas.